SN TRANSPORT ON ACCELERATORS

DOE CoE Portability Workshop 4/19/16

Steven Rennich, NVIDIA

David Appelhans, IBM

Leopold Grinberg, IBM

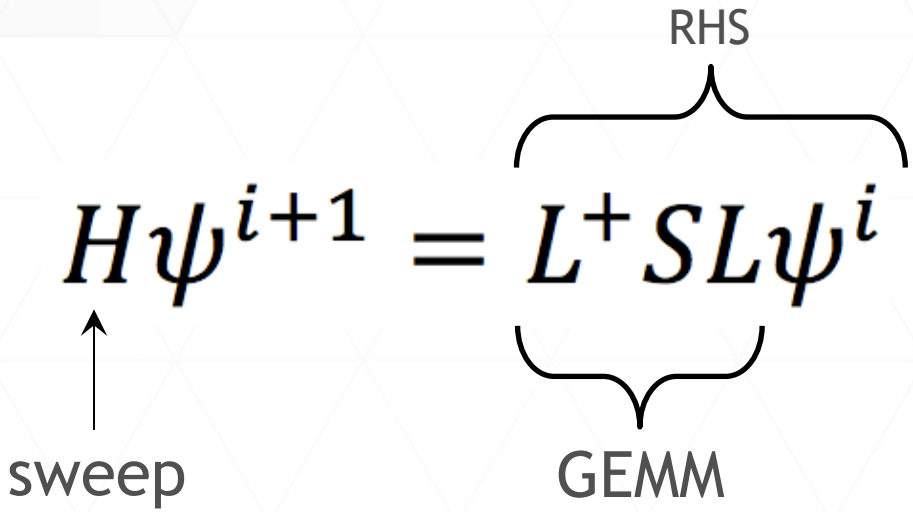
Adam Kunen, LLNL

others …

SN TRANSPORT

- Important class of DoE codes
  - Kripke/Ardra
  - UMT/Teton/Kull
  - **Sweep algorithm** – dominates performance at scale
- **GPU optimization of Diamond-Difference Sweep on Uniform Mesh**
  - **“Flux-register” or “Tiled-Hyperplane”**

SN TRANSPORT

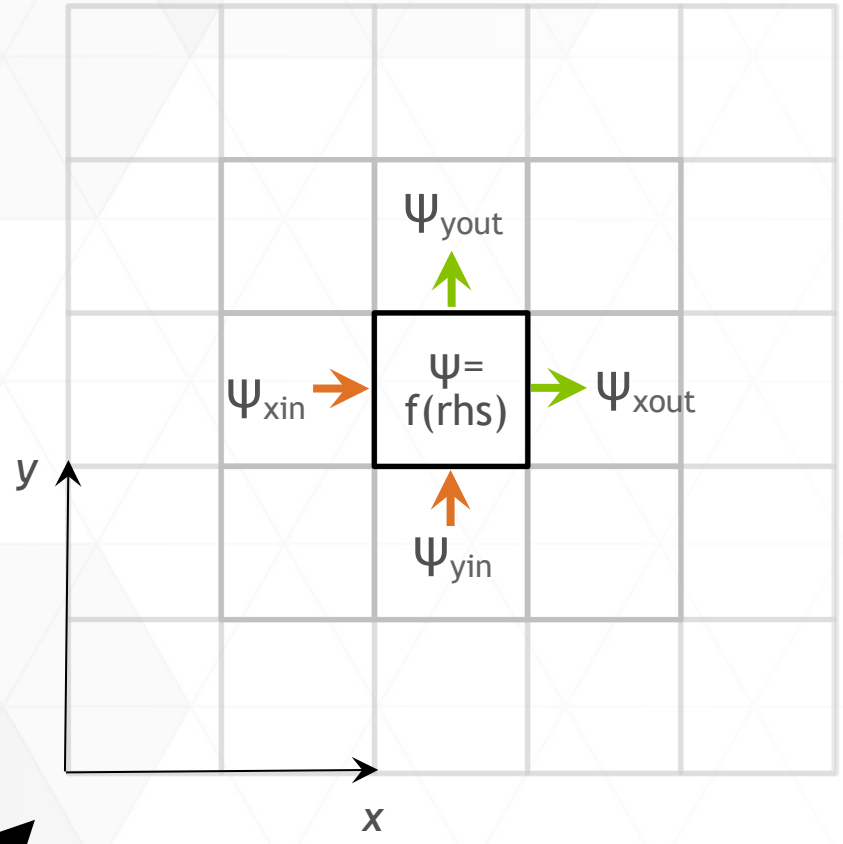


focus

easy – use batched algorithms

Kripke's version, in discrete form

DIAMOND DIFFERENCE SWEEP



$$C^* = \frac{\sigma_{t,g,c}}{2} + \frac{\mu_k}{\Delta x_c} + \frac{\eta_k}{\Delta y_c} + \frac{\xi_k}{\Delta z_c}$$

$$\psi_{zone} = \left[\frac{\mu_k}{\Delta x_c}\psi_{xin} + \frac{\eta_k}{\Delta y_c}\psi_{yin} + \frac{\xi_k}{\Delta z_c}\psi_{zin} + \frac{1}{2}Q_{zone}\right]\frac{1}{C^*}$$

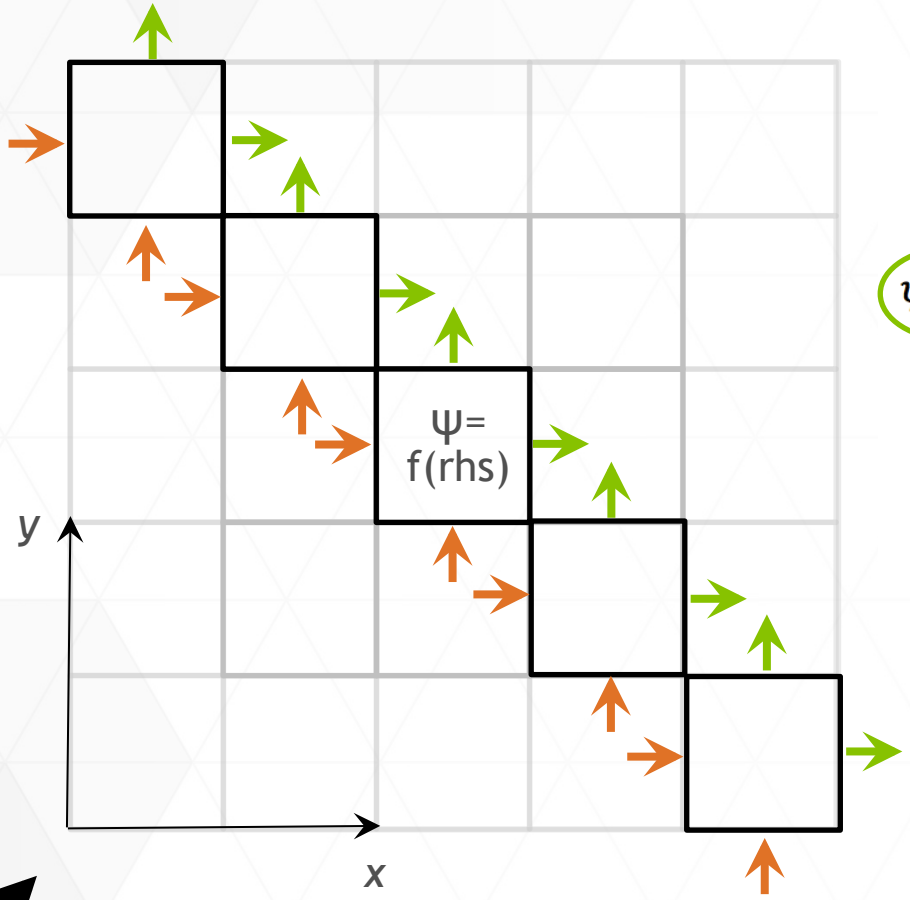
$$\psi_{xout} = 2\psi_{zone} - \psi_{xin}$$

$$\psi_{yout} = 2\psi_{zone} - \psi_{yin}$$

$$\psi_{zout} = 2\psi_{zone} - \psi_{zin}$$

ψ = flux

DIAMOND DIFFERENCE SWEEP



$$C^* = \frac{\sigma_{t,g,c}}{2} + \frac{\mu_k}{\Delta x_c} + \frac{\eta_k}{\Delta y_c} + \frac{\xi_k}{\Delta z_c}$$

$$\psi_{zone} = \left[\frac{\mu_k}{\Delta x_c}\psi_{xin} + \frac{\eta_k}{\Delta y_c}\psi_{yin} + \frac{\xi_k}{\Delta z_c}\psi_{zin} + \frac{1}{2}Q_{zone}\right]\frac{1}{C^*}$$

$$\psi_{xout} = 2\psi_{zone} - \psi_{xin}$$

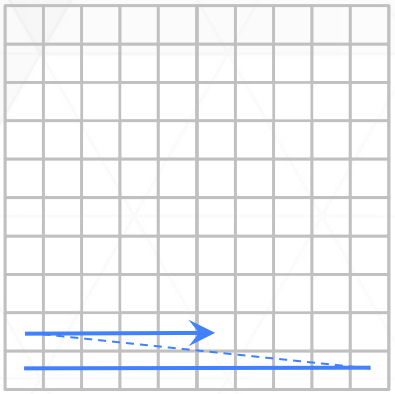
$$\psi_{yout} = 2\psi_{zone} - \psi_{yin}$$

$$\psi_{zout} = 2\psi_{zone} - \psi_{zin}$$

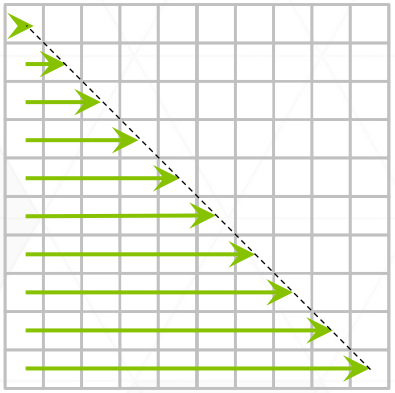
ψ = flux

SWEEP

KBA



Hyperplane



single-thread (per direction-group)
- cannot fill the GPU

no synchronization required

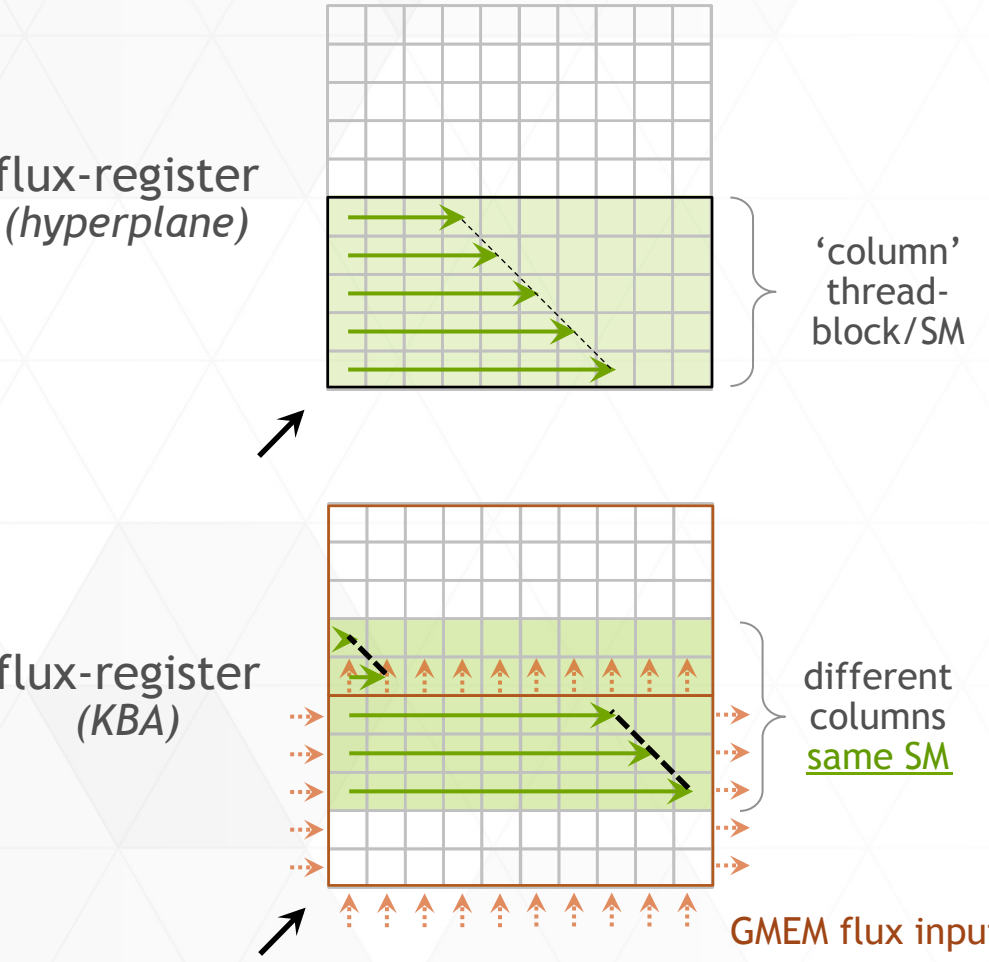
bandwidth bound

multiple threads (per direction-group)
- can fill the GPU

synchronization at every step
- expensive across SMs (kernel launch)

bandwidth bound
- all fluxes need to be read/written from GMEM every step.
- (8 numbers – RHS, Psi, 3 input/output fluxes)

GPU FLUX-REGISTER SWEEP



"Keep the fluxes in registers"
- x-fluxes in registers
- y-, z-fluxes in shared memory (SMEM)
- reduce GMEM bandwidth
- reduces bandwidth by ~4x (8->2, just RHS, Psi
- limited to within an SM (i.e. SMEM)

Synchronize only within the SM
- __syncthreads()
- very fast
- limited to within an SM (**threadblock**)

1 SM/block sweeps entire local domain (xd-g)

'Column' boundary data written to GMEM

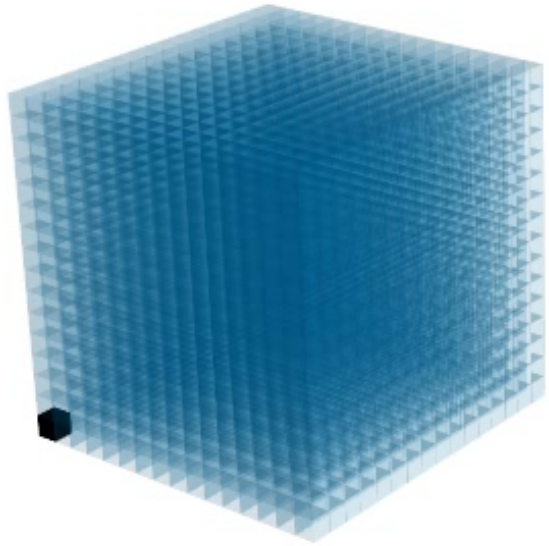
GPU FLUX-REGISTER SWEEP

(Visualization: David Appelhans IBM)

16x16x16 zones local domain

8x8 y-z ‘column base’

4 - columns



GPU FLUX-REGISTER SWEEP

- 8x8 y-z column base x 16 directions
  - Limited by registers
  - 53 registers per thread (also cache geometry/etc. in registers)
  - 16 kB of SMEM (of 48kB)
  - 50% occupancy
- 16x16 y-z column base version (4 directions)
  - ¼ of data swept, ~½ sweep latency, ~½ throughput
  - Options
- Sweep 15 groups simultaneously (on K40)
  - 15 SMs, each sweeping 16 directions and 1 group -> 16 blocks/groups per kernel
  - Limiting kernels to 15 blocks permits synchronization with MPI transfer of fluxes

GPU FLUX-REGISTER PERFORMANCE

- Typical configuration:
  - 32 x 32 x 32 local zones, 16 directions per octant, *120* groups (~6GB on GPU)
- *Relative Performance: Single node,* full-app (incl. batched DGEMM)
  - GPU flux-register performance = **7.2x** vs. CPU (1x K40)
  - GPU flux-register performance = **11x** vs. CPU (**2x K40**)
- ***Absolute Performance:*** Single node
  - Sweep latency = 1.8 ms [or 800 usec]
  - 58% SOL (theo. eff.) for GMEM bandwidth (110 GB/s vs. 190 GB/s peak *achieved*)
  - Opportunity to be 72% faster (Optimizations Remain!)

PORTABILITY

- Algorithmic portability
- “In fact, the KBA sweep is just the flux register sweep with a subdomain size of 1x1 and the hyperplane sweep is the flux register sweep with a subdomain size of jmax x kmax. But how the algorithm maps this to threads is fundamental to the performance difference between these identical mathematical formulations, and thus is paramount to the portability.”
- Portable performant *code* needs to start with a portable performant *algorithm*

PORTABILITY

- *Mapping algorithm to architecture*
- GPU implementation
  - Blocks operations to promote coalesced memory access (4 or 16 directions/groups) (memory architecture)
  - Use registers for x-flux (memory architecture)
  - Maximizes cooperative threadblock size
    - Maximize occupancy (processor architecture)
    - Use Shared Memory for y- and z-flux (memory architecture)
    - Synchronize h-planes within threadblock (processor architecture)
  - Size subdomains to fill SMs (processor architecture)

TILED HYPERPLANE SWEEP OPENMP4

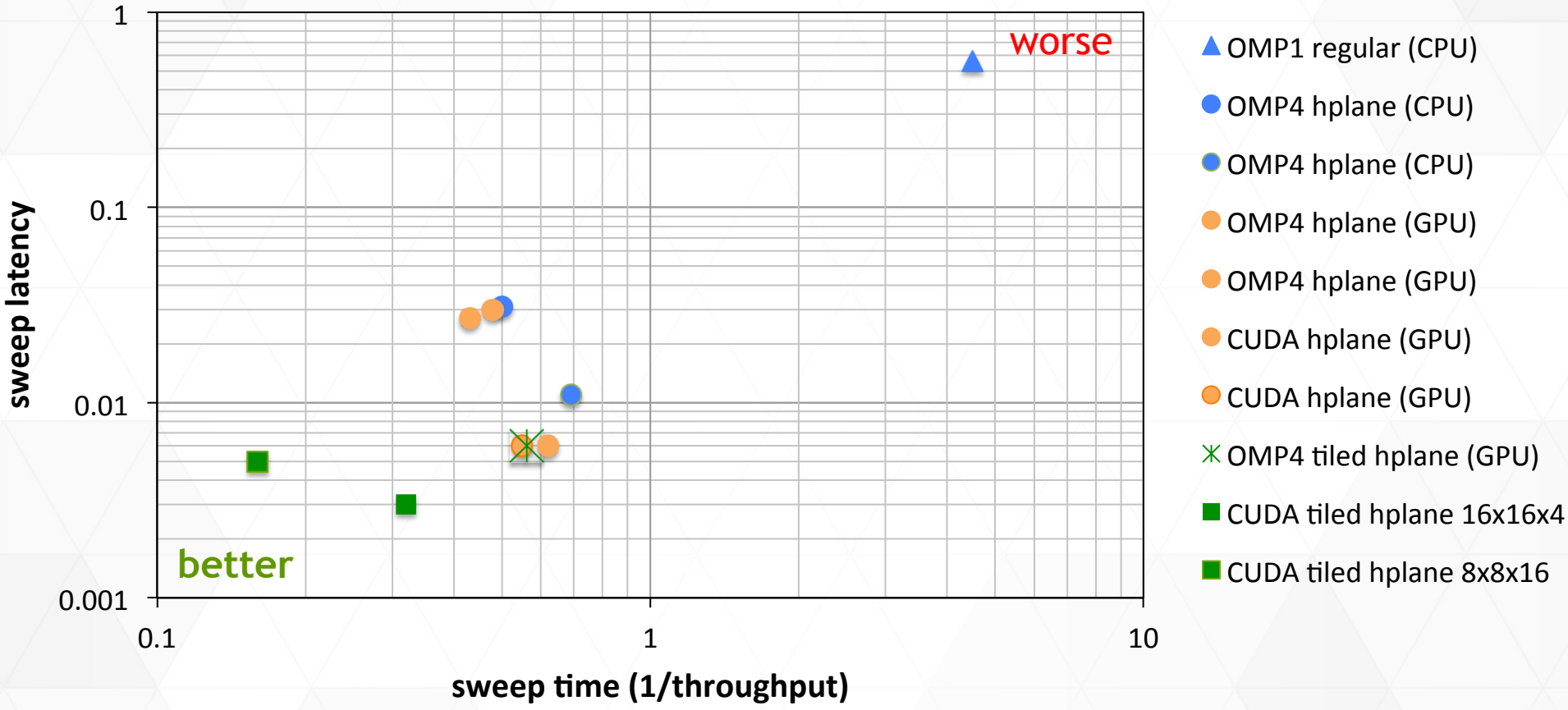
- **LLVM OpenMP4 compiler is young - excessive register allocation problem limits number of threads (for no spilling).**
  - 162 registers/thread for no spilling.
  - 256 threads per block, 4 kb of shared memory per block.
  - 3 blocks per SM on K80 (register limited)
  - Theoretical max occupancy: 37.5%.
  - 8x8 subdomain of zones, 4 directions per zone.

SWEEP COMPARISON

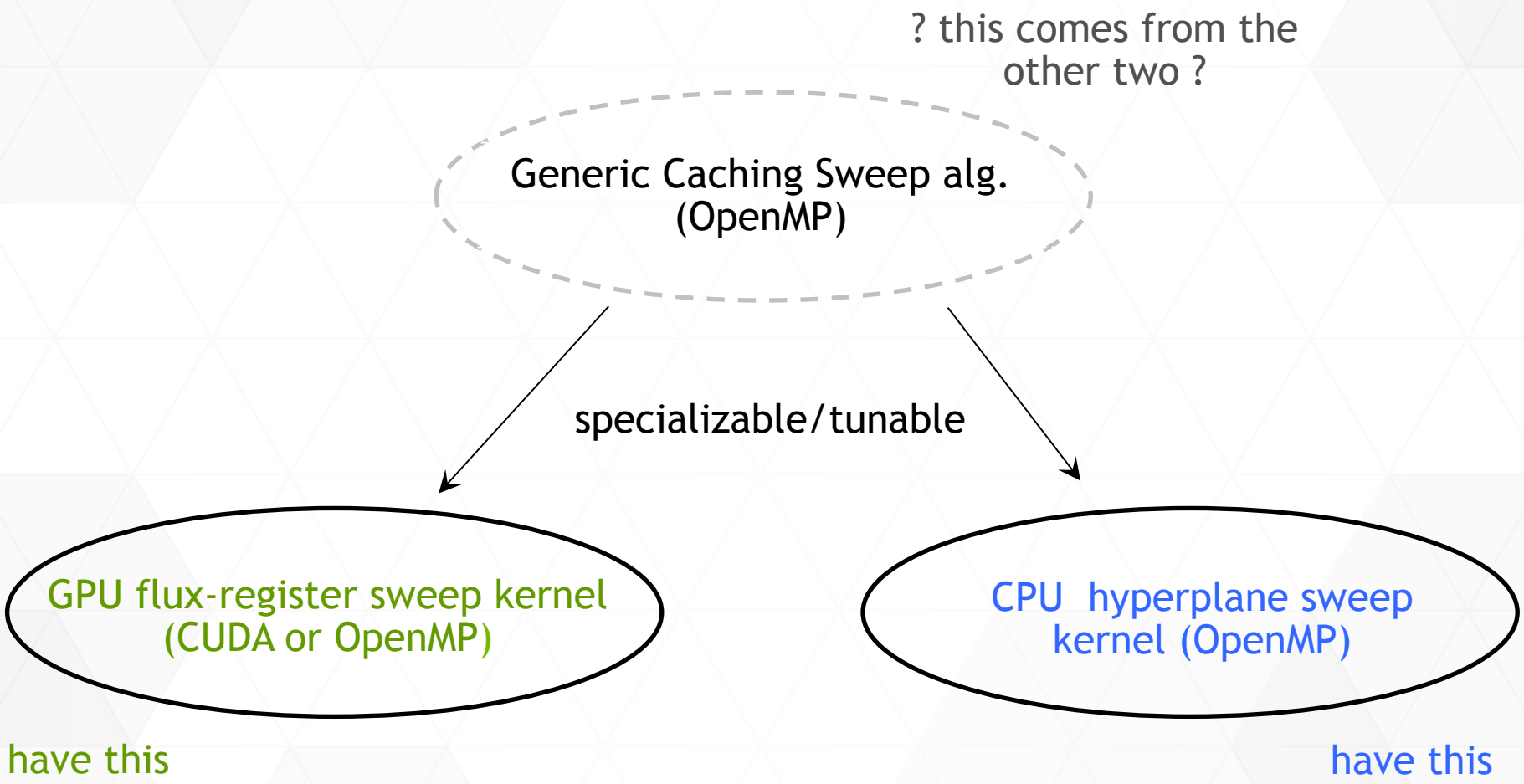
Units: ms/million dof	(normalized) Sweep Time	Sweep Latency
OMP1 regular (CPU)	4.51	0.563
OMP4 hplane (CPU)	0.50	0.031
OMP4 hplane (CPU)	0.69	0.011
OMP4 hplane (GPU)	0.48	0.030
OMP4 hplane (GPU)	0.62	0.006
CUDA hplane (GPU)	0.43	0.027
CUDA hplane (GPU)	0.55	0.006
OMP4 tiled hplane (GPU)	0.56	0.006
CUDA tiled hplane 16x16x4	0.32	0.003
CUDA tiled hplane 8x8x16	0.16	0.005

Table: Table of sweep times for 32x32x32 zone domain, 96-128 directions per octant, 104-156 groups. Multiple timings represent different group/direction set choices. CPU was dual socket, 20 core POWER 8, all GPU results were for a single K80 (half a physical card).

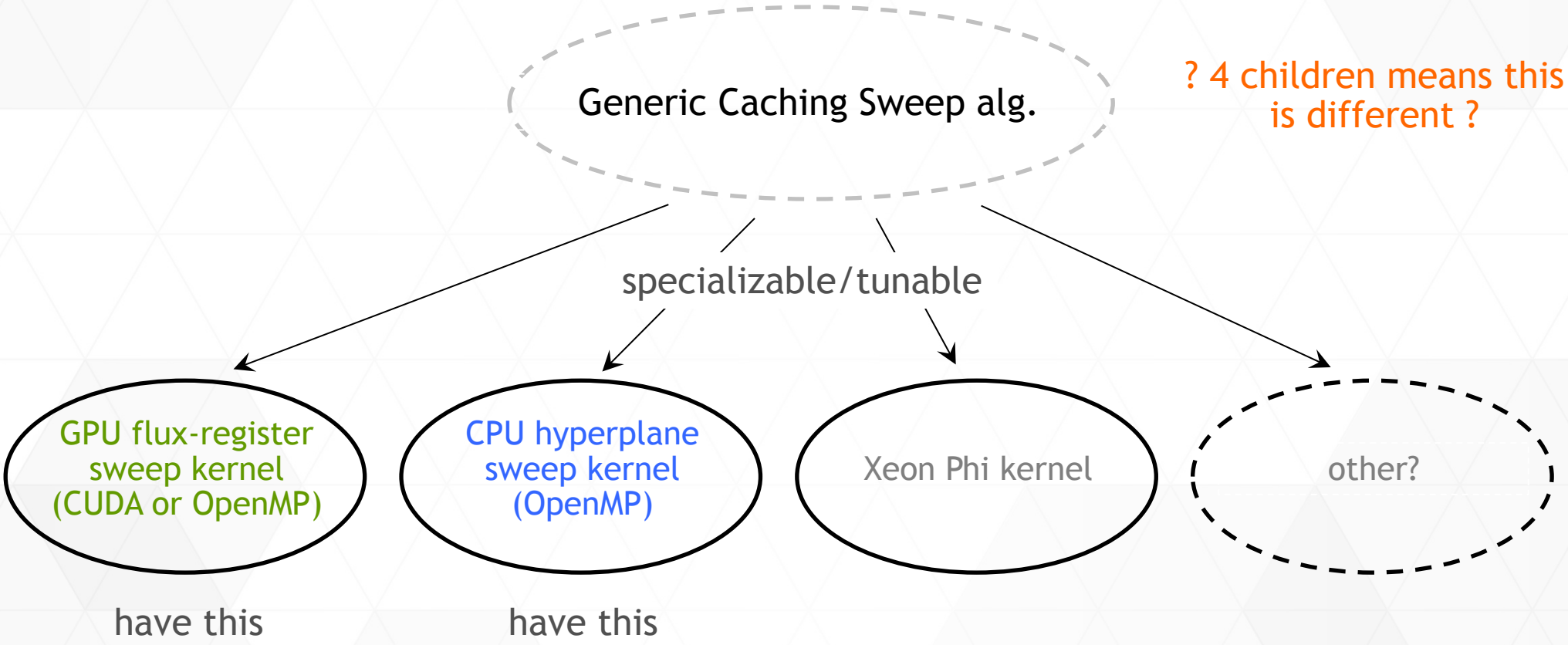
SWEEP TIMINGS



PORTABLE, PERFORMANT ALGORITHMS



PORTABLE, PERFORMANT ALGORITHMS



SUMMARIZING

- Optimized sweep kernel written in CUDA
  - Leverages many architectural features of the GPU
- OpenMP4 port
  - Performs well on CPU & GPU
  - Less performance than CUDA version on GPU
  - Expect better performance when compilers use fewer registers
- Performance portability requires
  - Performance portable algorithm
  - Portable language exposing architectural features